Ansible Collection for OpenAFS

Sine Nomine Associates

Jan 24, 2023

CONTENTS:

This is an Ansible collection to deploy the OpenAFS distributed filesystem. The collection provides roles, modules, and example playbooks to install and configure OpenAFS clients and servers.

CHAPTER

ONE

INTRODUCTION

The OpenAFS Ansible Collection is a collection of roles, plugins, and example playbooks to deploy and manage OpenAFS clients and servers with Ansible.

OpenAFS may be installed from pre-built packages or installed from source code. Ansible modules are provided to create OpenAFS volumes and users after the servers and at least one client has been installed.

Since OpenAFS requires Kerberos for authentication, roles are provided to deploy a Kerberos 5 realm with MIT Kerberos. Alternatively, an existing realm can be used for authentication.

1.1 Platforms supported

- AlmaLinux 8, 9
- CentOS 6, 7, 8
- Debian 10, 11
- Fedora 36, 37
- FreeBSD 12, 13
- openSUSE 15
- OracleLinux 8, 9
- Red Hat Entrerprise Linux 7, 8, 9
- Rocky 8, 9
- Solaris 11.4
- Ubuntu 20, 22

CHAPTER

TWO

INSTALLATION

2.1 Requirements

This collection requires Ansible 2.10 or later. Ansible may be installed with your package manager or with Python `pip`. A Python virtualenv is recommended if you are installing packages with `pip`.

The target machines to be managed must be reachable via `ssh` and must have Python already installed. Ansible is not required on the target machines.

2.2 Galaxy

The **OpenAFS Ansible Collection** is available on Ansible Galaxy. Install the collection with the `ansible-galaxy` command:

```
$ ansible-galaxy collection install openafs_contrib.openafs
```

Use the `--force` option to overwrite the currently installed version to upgrade if you already have an older version of the collection installed.

2.3 Source

Install the **OpenAFS Ansible Collection** from source code with the following commands:

```
$ cd <your-project-directory>
$ mkdir -p ansible_collections/openafs_contrib
$ cd ansible_collections/openafs_contrib
$ git clone https://github.com/openafs-contrib/ansible-openafs openafs
$ cd openafs
$ make init
$ source .venv/bin/activate
$ make install
```

The directory structure shown above is required for proper operation of the molecule unit tests and document generation.

CHAPTER

THREE

USAGE

3.1 Installation Methods

OpenAFS may be installed and updated with prebuilt packages or built from source on the remote nodes. The following installation methods are supported:

- **managed** - Install with the distro's package manager (e.g., `yum`, `apt`).
- **packages** - Install prebuilt packages (e.g. `rpm`, `dpkg`).
- **bdist** - Install a binary distribution with Transarc-style paths.
- **source** - Install from source code.
- **none** - Skip installation tasks.

Different installation methods may be used on each remote node. However, when installing a client and server on the same node, the same installation method must be specified for the client and server roles, that is, you cannot mix installation methods on a given node.

The installation method is stored id the `/etc/ansible/facts.d/openafs.fact` json file on the remote node. This file must be changed if you want to change the installation method after OpenAFS has already been installed.

3.2 Inventory

Provide an Ansible inventory file for your host configuration. Ansible supports `ini` and `yaml` inventory files.

The **OpenAFS** ansible roles and example playbooks use the following group name conventions:

afs_kdcs The Kerberos KDC server.

> Currently, only one KDC is supported. Additional KDCs can be deployed with custom playbooks or community supported roles.

afs_databases The OpenAFS database servers hosts.

afs_fileservers The OpenAFS fileserver hosts.

afs_clients The OpenAFS clients hosts.

afs_admin_client The OpenAFS client host used for initial cell configuration.

Note that a given host may be a member of more than one group. For example, a given host can be in the `afs_databases`, `afs_fileservers`, and the `afs_clients` groups.

3.2.1 Example Inventory

```
[afs_kdcs]
kdc

[afs_databases]
db1
db2
db3

[afs_fileservers]
fs01
fs02
fs03
fs04

[afs_clients]
client[01:20]

[afs_admin_client]
client01

[afs_cell:children]
afs_databases
afs_fileservers
afs_clients

[afs_cell:vars]
afs_realm = EXAMPLE.COM
afs_cell = example.com

[afs_clients:vars]
afs_kdc_servers = kdc
afs_kadmin_server = kdc
afs_module_install_method = dkms
```

3.3 Cell Configuration (CellServDB)

The OpenAFS cell configuration (CellServDB file) is provided as an inventory variable or an external yaml file (with the same structure as the inventory variable.) The cell configuration contains the list of database server IPv4 addresses.

To specify the cell configuration with an inventory variable, add the `afs_csdb` dictionary to your inventory for all of the hosts in your cell. If your inventory is in `ini` format, then provide a `afs_cell.yaml` file in the Ansible `group_vars` directory.

```
# Contents of `group_vars/afs_cell.yaml`
afs_csdb:
  cell: example.com
  desc: My Example Cell
  hosts:
    - ip: 192.168.122.219
```

(continues on next page)

(continued from previous page)

```
      name: afs02
      clone: no
    - ip: 192.168.122.154
      name: afs03
      clone: no
    - ip: 192.168.122.195
      name: afs04
      clone: no
```

A `csdb.yaml` file can be generated from a playbook and then saved for later use. This can be especially useful when creating short lived test cells from newly created virtual machine clusters.

```
# Retrieve the addresses of the database servers and generate
# a cell configuration yaml file (csdb.yaml)
- name: Create CellServDB
  hosts: afs_databases
  tasks:
    - include_role:
        name: openafs_contrib.openafs.openafs_common
        tasks_from: generate_csdb
      when: afs_csdb is undefined
```

3.4 Running playbooks

Create a set of Ansible playbooks for your environment to deploy the OpenAFS servers and clients. See the example playbooks in the `playbooks` directory as a starting point.

Run the playbooks with `ansible-playbook [options] <playbooks>`.

Import the `openafs_client` role to install and configure client machines, and import the `openafs_server` role to install and configure fileserver and database server machines. A single machine may have both a client and server installed on it, but with the limitation the client and server versions must match.

Use the `openafs_volume` module on a client machine to create and mount the OpenAFS `root.afs` and `root.cell` volumes. This module may also be used to create additional volumes.

Use the `openafs_user` module on a client to create initial users.

See the Ansible documentation for more information on running `ansible-playbook`.

CHAPTER

FOUR

PLAYBOOKS

The following playbooks are provided as starting points for your playbooks. The playbooks show how the collection roles and plugins can be used to deploy a Kerberos realm and an OpenAFS cell. See the `playbooks` directory.

- *build.yaml* Build OpenAFS binaries
- *realm.yaml* Install and setup a Kerberos realm
- *cell.yaml* Install and setup an OpenAFS cell

CHAPTER

FIVE

ROLES

5.1 openafs_client - OpenAFS Client Role

5.1.1 Description

Install and configure OpenAFS clients.

5.1.2 Requirements

Unless DNS SRV records have been configured to supply the OpenAFS database server addresses, the names and addresses of the OpenAFS databases to setup the server CellServDB files must be provided by the `afs_csdb` inventory variable, or a separate yaml file, the path of which is specified by the `afs_csdb_file` variable.

5.1.3 Variables

See `openafs_common` for included common variables.

afs_module OpenAFS kernel module name, `openafs` or `libafs`. Default: `openafs`

afs_module_install_method Specifies the kernel module installation method on RPM-based systems. Must be one of: `dkms`, `kmod`, or `kmp`.

Default: `dkms`

afs_module_enable_preload Specifies if the role should attempt to preload the OpenAFS module before the client service is started.

Default: no

afs_mountpoint The AFS filesystem mount point. This value is written to the `cacheinfo` file.

Default value is autodetected from the `cacheinfo` file installed by packages.

Default: autodetected, fallback to `/afs`

afs_cachedir The path to the AFS cache. This value is written to the `cacheinfo` file.

Default value is autodetected from the `cacheinfo` file installed by packages.

Default: autodetected, fallback to `/usr/vice/cache`

afs_cachesize The size of the AFS cache. This valus is written to the `cacheinfo` file.

Default value is autodetected from the `cacheinfo` file installed by packages.

Default: autodetected, fallback to 50000

afs_afsd_opts The `afsd` command line arguments to override the value provided by the client installation package.

Default: auto-detected

afs_client_netinfo A single string, or a list of strings, to set the contents of the client `NetInfo` configuration file.

afs_client_netrestrict A single string, or a list of strings, to set the contents of the client `NetRestrict` configuration file.

5.2 openafs_common - OpenAFS Common Role

5.2.1 Description

Common definitions for OpenAFS clients and servers.

5.2.2 Variables

afs_cell The OpenAFS cell name. Default: `example.com`

afs_realm The Kerberos realm name. Defaults to the uppercased cell name. Default: `EXAMPLE.COM`

afs_cell_files Location of cell specific files on the controller. Default: `~/.ansible_openafs/cell/<cell>`

afs_csdb The CellServDB information for this cell. Undefined by default. The `afs_csdb` should be provided in your inventory. If not defined, the `afs_csdb` is read from the external yaml file located at the fully qualified path given by the variable `afs_csdb_file`.

```
afs_csdb:
  cell: example.com
  desc: Cell name
  hosts:
    - ip: 192.168.122.219
      name: afs02
      clone: no
    - ip: 192.168.122.154
      name: afs03
      clone: no
    - ip: 192.168.122.195
      name: afs04
      clone: no
```

afs_csdb_file The path to the external yaml file containing CellServDB information for the cell. This file is read when the `afs_csdb` is not defined in the inventory. The `afs_csdb_file` can be created in a playbook with the `generate_csdb` task. This can be useful in to automatically create a usable CellServDB file in a test environment.

The CellServDB information for the cell. This must be provided as a inventory variable or an external yaml file, the path specified by `afs_csdb_file`.

Default: `<afs_cell_files>/csdb.yaml`

afs_admin An administrative user name. This is the `pts` user name, for example: `jdoe.admin` Can also be a list. Default: `<ansible_user>.admin`

afs_install_method The method used to install the OpenAFS client and/or server software on this remote node. Must be one of: `managed`, `packages`, `bdist`, `source`.

Default: platform dependent

afs_checkout_method The method used to checkout source code when the `afs_install_method` is `source`. Must be one of: `git`, `sdist_upload`, `gerrit`.

Default: `git`

afs_yum_repo The yum repo url when `afs_install_method` is `managed`.

Default: `https://download.sinenomine.net/openafs/rpms/el$releasever/$basearch`

afs_install_archive Path to the compressed archive containing the installation files when the `afs_install_method` is `packages`, `bdist`, or `sdist`.

afs_git_repo The git repository URL when the `afs_install_method` is `source`.

Default: `git://git.openafs.org/openafs.git`

afs_git_version The git branch or tag to check out and build when the `afs_install_method` is `source`.

Default: `master`

afs_gerrit_host The gerrit hostname when `afs_checkout_method` is `gerrit`.

Default: `gerrit.openafs.org`

afs_gerrit_number The gerrit number to be fetched when `afs_checkout_method` is `gerrit`. The most recent patch-set is fetched. This variable is manditory when `afs_checkout_method` is `gerrit`.

afs_configure_options Overrides the options given to configure when building from source. This variable can be a simple string, such as "`--enable-debug --enable-transarc-paths`", or may be specified as a dictionary, for example:

```
  afs_configure_options:
    prefix: /usr
    bindir: /usr/bin
    libdir: /usr/lib64
    sbindir: /usr/sbin
    disable:
      - strip_binaries
      - kernel_module
    enable:
      - debug
      - redhat_buildsys
      - transarc_paths
    with:
      - krb5: /path/to/krb5.lib

Default: detected, platform dependent
```

afs_nolibafs_configure_options Overrides the *configure* arguments whe building the userspace binaries from source (no kernel module). This variable can be a simple string, such as "`--enable-debug --enable-transarc-paths`", or may be specified as a dictionary.

Default: detected, platform dependent

afs_always_build When the `afs_install_method` is `sdist` or `source`, force a rebuild and reinstall even if a change in the source code checkout was not detected.

Default: no

afs_clean_build When the `afs_install_method` is `sdist` or `source`, clean any build artifacts that may be left from a previous build. Set to no to let make only rebuild binaries which are out of date with the sources, which should be faster when rebuilding the same branch as the previous build.

Default: yes

afs_transarc_build When the `afs_install_method` is `sdist` or `source`, build and install the OpenAFS binaries with the legacy Transarc style installation paths, e.g, `/usr/afs/bin`, `/usr/afs/etc`. This option will add `--enable-transarc-paths` to the `configure` options.

Default: no

5.3 openafs_devel - OpenAFS Development Role

5.3.1 Description

Install development packages needed to build OpenAFS userspace binaries.

5.3.2 Variables

afs_devel_oracle_key Fully qualified path of your Oracle Developer Studio key on the controller. Used to install Oracle Developer Studio when the remote node operating system is Solaris.

Default: `~/.certs/pkg.oracle.com.key.pem`

afs_devel_oracle_cert Fully qualified path of your Oracle Developer Studio certificate on the controller. Used to install Oracle Developer Studio when the remote node operating system is Solaris.

Default: `~/.certs/pkg.oracle.com.certificate.pem`

5.4 openafs_devel - OpenAFS Kernel Development Role

5.4.1 Description

Install development packages needed to build the OpenAFS kernel module

5.4.2 Variables

afs_devel_allow_kernel_update When the kernel headers cannot be installed for the currently running kernel version, automatically update the kernel and reboot.

Default: no

5.5 openafs_krbclient - Kerberos Client Role

5.5.1 Description

Install and configure Kerberos workstation packages.

5.5.2 Variables

afs_realm The Kerberos realm name.

Default: EXAMPLE.COM

afs_kdc_servers A comma separated list of kdc host names to be set in the krb5.conf file. If this variable is not defined, the hostname of the hosts in the `afs_kdcs` inventory group are used. If that group is not defined, the kdcs are not defined for the realm in the krb5.conf file and it is assumed they are defined as SRV records in DNS.

Default: hosts in the `afs_kdcs` group

afs_kadmin_server The host name of the kadmin server to be set in the krb5.conf file. If this variable is not defined, the first host name in the `afs_kdcs` inventory group is used. If that group is not defined, the kadmin server hostname is not set in the krb5.conf file and it is assumed to be defined as SRV records in DNS.

Default: undefined

afs_krb_dns_lookup_kdc Define if dns_lookup_kdc mode is enabled/disabled via true/false. If this variable is not defined, no entry will be set which is the same like dns_lookup_kdc = true.

Default: undefined

5.6 openafs-krbserver - Kerberos Server Role

5.6.1 Description

Install and configure the MIT Kerberos master KDC on single host, create the Kerberos database and the first administrator principal.

5.6.2 Variables

afs_realm The Kerberos realm name.

Default: EXAMPLE.COM

afs_krb_master_password The secret Kerberos database master password. The password is random by default. If a value is provided, it should be encrypted with `ansible-vault`.

Default: <random>

afs_krb_admin_principal A administrator principal to be created by this role. A keytab is always created for this principal, even when the password is also provided.

Default: root/admin

afs_krb_admin_password The admin principal password. This password random by default. If a value is provided, it should be encrypted with `ansible-vault`.

Default: <random>

afs_kdc_servers A comma separated list of kdc host names to be set in the krb5.conf file. If this variable is not defined, the hostname of the hosts in the `afs_kdcs` inventory group are used. If that group is not defined, the kdcs are not defined for the realm in the krb5.conf file and it is assumed they are defined as SRV records in DNS.

Default: hosts in the `afs_kdcs` group

afs_kadmin_server The host name of the kadmin server to be set in the krb5.conf file. If this variable is not defined, the first host name in the `afs_kdcs` inventory group is used. If that group is not defined, the kadmin server hostname is not set in the krb5.conf file and it is assumed to be defined as SRV records in DNS.

Default: undefined

afs_krb_dns_lookup_kdc Define if dns_lookup_kdc mode is enabled/disabled via true/false. If this variable is not defined, no entry will be set which is the same like dns_lookup_kdc = true.

Default: undefined

afs_krb_max_life KDC max ticket life.

Default: 10h 0m 0s

afs_krb_max_renewable_life KDC max renewable life.

Default: 7d 0h 0m 0s

afs_krb_supported_enctypes KDC supported enctypes. Specify as a list of enctype:salt values.

Default: [‘aes256-cts-hmac-sha1-96:normal’, ‘aes128-cts-hmac-sha1-96:normal’]

afs_krb_default_principal_flags KDC default principal flags.

Default: +preauth

5.7 openafs_server - OpenAFS Server Role

5.7.1 Description

Install and configure OpenAFS servers. This role installs both the fileserver and the database servers, which can be installed on the same hosts or different hosts.

This role configures the system to allow OpenAFS servers operate correctly in selinux enforcing mode when installing from RPM packages.

5.7.2 Requirements

A Kerberos realm is required before creating the OpenAFS services. This can be a pre-existing realm or can be created with the `openafs_krbserver` role. A service principal is required and must be exported to a keytab file. See the `realm.yml` example playbook.

The servers may be installed from the distribution package manager if packages are available, installed from pre-built binaries created by separate process or playbook (see the `openafs_devel` role and `build.yml` example playbook), or installed from source code from a git repository.

The names and addresses of the OpenAFS databases to setup the server CellServDB files must be provided by the `afs_csdb` inventory variable, or a separate yaml file, the path of which is specified by the `afs_csdb_file` variable.

5.7.3 Variables

afs_security_model The system security model. Should be `none` or `selinux`. When set to `selinux`, the selinux contexts for OpenAFS will be updated to allow the server to run with selinux enabled.

default: none

afs_is_fileserver Indicates the node is a fileserver. By default, `afs_is_fileserver` is true when the node is a member of the `afs_fileservers` group.

default: check `afs_fileservers` group

afs_is_dbserver Indicates the node is a database server. By default, `afs_is_dbserver` is true when the node is a member of the `afs_databases` group.

default: check `afs_databases` group

afs_fileserver_type Determines the fileserver type the node is a fileserver. Valid values are `fs` (legacy File Server) or `dafs` (modern Demand Attach FileServer).

Default: `dafs`

afs_server_cold_start Treat this play as the initial installation of the servers, in which case wait for the database servers to reach quorum before starting the fileservers. This avoids a 5 minute delay for the fileservers to retry registration with the VLDB.

Set to **yes** (True) to defer fileserver startup until database quorum is detected.

Set to **no** (False) to skip cold start checks and tasks.

Default is to detect by checking for the presence of the `BosConfig` file.

afs_pseudo_partitions The list of pseudo fileserver vice partitions to be created. Pseudo partitions are directories in the root partition, with the special `AlwaysAttach` file to indicate they should be attached by the fileserver. This feature is intended for testing. Specify the pseudo partitions by partition id, that is `a`, `b`, etc.

default: []

afs_create_root Ensure the `root.afs` and `root.cell` volumes exist. The `root.afs` volume must exist before starting clients without the `--dynroot` option. Modern clients typically are started with the `--dynroot` option and so are able to start without the presences of the root volumes.

default: yes

afs_server_netinfo A single string (or list of strings) to set the contents of the server `NetInfo` configuration file. This file specifies which addresses or subnetworks should be used for server communication. A specific address can be forced by specifying a `f` prefix.

afs_server_netrestrict A single string, or a list of strings, to set the contents of the server `NetRestrict` configuration file. This file specifies which addresses or subnetworks should be excluded from server communications.

afs_service_keytab The AFS service Kerberos keytab file. This is the file path of the keytab file on the controller, which should be protected with `ansible-vault`. The keytab file will be uploaded to the server nodes unless `afs_service_keytab_externally_managed` is true. The keys will be imported with the `akeyconvert` tool on servers running OpenAFS 1.8.x (or greater). The uploaded keytab file will be named `rxkad.keytab` for compatibility with OpenAFS 1.6.x.

default: <afs_cell_files>/afs.<afs_cell>.keytab

afs_service_keytab_externally_managed When true, the AFS service Kerberos keytab file is managed with an external secrets management tool.

default: false

afs_bosserver_restricted_mode Run the BosServer in restricted mode. This mode improves the security of the BosServer by prohibiting bos commands which are not needed for routine operation.

The following bos commands are not available when the BosServer running in restricted mode: `bos exec`, `bos uninstall`, `bos install`, `bos create`, `bos delete`, `bos prune`, and the `bos getlog` is limited to server log files.

default: yes

afs_bosserver_bnodes Extra `bnode` entries to add to `BosConfig`.

default: []

example:

```
afs_bosserver_bnodes:
  - name: backup
    type: cron
    goal: 1
    parm:
      - /usr/afs/backup/clones/lib/backup.csh daily
      - 05:00
```

afs_bosserver_opts The `bosserver` command line options.

afs_ptserver_opts The `ptserver` command line options.

afs_vlserver_opts The `vlserver` command line options.

afs_dafileserver_opts The `dafileserver` command line options.

afs_davolserver_opts The `davolserer` command line options.

afs_salvageserver_opts The `salvageserver` command line options.

afs_dasalvager_opts The `dasalvager` command line options.

afs_fileserver_opts The `fileserver` command line options.

afs_volserver_opts The `volserver` command line options.

afs_salvager_opts The `salvager` command line options.

CHAPTER

SIX

MODULES

6.1 openafs_build – Build OpenAFS binaries from source

6.1.1 Synopsis

Build OpenAFS server and client binaries from source code by running `regen.sh`, `configure`, and `make`. The source code must be already present in the *srcdir* directory.

The *openafs_build* module will run the OpenAFS `regen.sh` command to generate the `configure` script when the `configure` script is not already present in the *srcdir*.

Unless the *configure_options* option is specified, the configure command line arguments are determined automatically, based on the platform and *openafs_build* options.

The `make` program is run to build the binaries. Unless the *target* options is specified, the make target is determined automatically.

A complete set of build log files are written on the *logdir* directory on the host for build troubleshooting.

Out-of-tree builds are supported by specifying a build directory with the *builddir* option.

`git clean` is run in the *srcdir* when *clean* is true and a `.git` directory is found in the `srcdir`. When *clean* is true but a `.git` directory is not found, then `make clean` is run to remove artifacts from a previous build. When *clean* is true and an out-of-tree build is being done, all of the files and directories are removed from the *builddir*.

An installation file tree is created in the *destdir* directory when the *target* starts with `install` or `dest`. The files in *destdir* may be installed with the *openafs_install_bdist* module.

See the `openafs_devel` role for tasks to install required build tools and libraries on various platforms.

6.1.2 Requirements

The below requirements are needed on the host that executes this module.

- tools and libraries required to build OpenAFS

6.1.3 Parameters

srcdir (True, path, None) Source files must have been previously checkout or copied to this path.

builddir (optional, path, <srcdir>) The path for out-of-tree builds.

logdir (optional, path, <srcdir>/.ansible) The path to store build log files.

The logdir may be a subdirectory of the `srcdir`.

The logdir may not be a subdirectory of the `builddir` when doing an out-of-tree build.

destdir (optional, path, <srcdir>/packages/dest) The destination directory for `install` and `dest` targets and variants.

The tree staged in this directory may be installed with the *openafs_install_bdist* module.

clean (optional, bool, False) Run `git clean` in the *srcdir* when it contains a `.git` directory, otherwise run `make clean`.

Remove the *builddir* when using an out of tree build, that is the *builddir* is different than the *srcdir*.

A *clean* build should be done to force a complete rebuild.

The *clean* option will remove any new files you added manually on the remote node and did not commit when the *srcdir* is a git repository.

make (optional, path, detect) The `make` program to be executed.

jobs (optional, int, the number of CPUs on the system) Number of parallel make processes.

Set this to 0 to disable parallel make.

build_manpages (optional, bool, True) Generate the man pages.

build_userspace (optional, bool, True) Build userspace programs and libraries.

build_module (optional, bool, True) Build the OpenAFS kernel module.

build_terminal_programs (optional, bool, True) Build curses-based terminal programs.

build_bindings (optional, bool, True) Build program language bindings with swig.

build_fuse_client (optional, bool, True) Build fuse client.

with_version (optional, str, None) Version string to embed in program files.

The *version* will be written to the `.version` file, overwritting the current contents, if any.

with_transarc_paths (optional, bool, False) Build binaries which use the legacy Transarc-style paths.

with_debug_symbols (optional, bool, True) Include debug symbols and disable optimizations.

with_rxgk (optional, bool, False) Include rxgk support.

configure_options (optional, raw, None) The explicit `configure` command arguments. When present, this option overrides the `build_*` and `with_*` options.

May be specified as a string, list of strings, or a dictionary.

When specified as a dictionary, the values of the keys `enabled`, `disabled`, `with`, and `without` may be lists.

configure_environment (optional, dict, None) Extra environment variables to be set when running `configure`.

target (optional, str, detect) The make target to be run.

The make target will be determined automatically when this option is omitted.

6.1.4 Examples

```
- name: Build OpenAFS from source
  openafs_contrib.openafs.openafs_build:
    srcdir: ~/src/openafs

- name: Build OpenAFS binaries for the current system.
  openafs_contrib.openafs.openafs_build:
    srcdir: ~/src/openafs
    clean: yes

- name: Build OpenAFS legacy distribution
  openafs_contrib.openafs.openafs_build:
    srcdir: ~/src/openafs
    clean: yes
    with_transarc_paths: yes

- name: Build OpenAFS server binaries with custom install paths.
  openafs_contrib.openafs.openafs_build:
    srcdir: ~/src/openafs
    clean: yes
    target: install_nolibafs
    destdir: packages/dest
    configure_options:
      prefix: /usr
      bindir: /usr/bin
      libdir: /usr/lib64
      sbindir: /usr/sbin
      disable:
        - strip_binaries
        - kernel_module
      enable:
        - debug
        - redhat_buildsys
        - transarc_paths
      with:
        - krb5: /path/to/krb5.lib
      with_linux_kernel_packaging: true
      with_swig: true
```

6.1.5 Return Values

srcdir (always, string, /home/tycobb/projects/myproject) Absolute path to the project directory.

builddir (always, string, /home/tycobb/projects/myproject) Absolute path to the build directory

destdir (when destdir is specified, string, /home/tycobb/projects/myproject/packages/dest) Absolute path to the installation files.

logdir (, string, /home/tycobb/projects/myproject/.ansible) Absolute path to the log files. May be used for *openafs_install_bdist*.

logfiles (always, list, ['/tmp/logs/build.log', '/tmp/logs/make.out', '/tmp/logs/make.err']) Log files written for troubleshooting

kmods (success, list, ['/home/tycobb/projects/myproject/src/libafs/MODLOAD-5.1.0-SP/openafs.ko']) The list of kernel modules built, if any.

6.1.6 Status

- This module is not guaranteed to have a backwards compatible interface. *[preview]*
- This module is maintained by community.

Authors

- Michael Meffie

6.2 openafs_build_packages – Build OpenAFS installation packages

6.2.1 Synopsis

Build OpenAFS installation packages from an OpenAFS source distribution.

This module supports building RPM packages for RedHat family distributions. Other packaging types may be added in the future.

The source distribution files must be already present in the *sdist* directory on the remote node. The source distribution files may be created with the `openafs_build_sdist` module.

The *openafs_build_packages* module will create the rpm workspace directories and populate the SPECS and SOURCES directories from the source distribution files and the file options, then will build the source and binary rpm files with `rpmbuild`.

The RPM package version and release strings are generated from the OpenAFS version string extracted from the `.version` file in the source archive.

See the `openafs_devel` role for tasks to install the required build tools and libraries.

6.2.2 Requirements

The below requirements are needed on the host that executes this module.

- Tools and libraries required to build OpenAFS.
- The `kernel-devel` package, when building the kernel module.
- `rpmbuild` tool

6.2.3 Parameters

build (optional, str, all) Specifies which packages to build.

`all` build source and binary RPMs for userspace and kernel module

`source` build the source RPM only

`userspace` build the source RPM and the userspace RPMs

`modules` build the source RPM and the kmod RPM

sdist (True, path, None) The path on the remote node to the source distribution files directory on the remote node.

The *sdist* directory must contain the `openafs-<version>-src.tar.bz2` source archive and the `openafs-<version>-doc.tar.bz2` documentation archive.

The *sdist* directory may also contain the `ChangeLog` file and the `RELNOTES-<version>` file.

spec (optional, str, None) The path on the remote node to a custom `openafs.spec` file to be used to build the rpm files. The `openafs.spec` file will be extracted from the source archive file when the *spec* option is not provided.

relnotes (optional, str, None) The path on the remote node to a custom `RELNOTES` file to be included in the build.

The `RELNOTES-<version>` in the *sdist* directory will be used when the *relnotes* option is not specified. The `NEWS` file will be extracted from the source archive if the `RELNOTES-<version>` file is not found in the *sdist* directory.

changelog (optional, str, None) The path on the remote node to a custom `ChangeLog` file to be included in the build.

The `ChangeLog` in the *sdist* directory will be used when the `changelog` option is not specified. An empty `ChangeLog` file will be created if the `ChangeLog` is not found in the *sdist* directory,

csdb (optional, path, None) The path on the remote node to a custom `CellServDB` file to be incuded in the build.

The `CellServDB` file in the *sdist* directory will be used when the *csdb* option is not specified. The `CellServDB` file will be extracted from the source archive if the `CellServDB` file is not found in the *sdist* directory.

patchdir (optional, path, I(sdist)) The path on the remote node of the directory containing patch files to be applied.

Patch names are identified by the `PatchXX` directives in the spec file.

kernvers (optional, str, current kernel version) The kernel version to be used when building the kernel module. By default, the kernel version of the running kernel will be used.

topdir (optional, path, C(~/rpmbuild)) The top level rpmbuild workspace directory on the remote node.

logdir (optional, path, I(topdir)/C(BUILD)) The path to write build log files on the remote node.

tar (optional, path, C(tar)) The `tar` program used to unpack the source archive.

tar_extra_options (optional, str, None) Extra command line options to unpack the source archive.

6.2.4 Examples

```
- name: "Checkout OpenAFS source code."
  git:
    repo: "git@openafs.org/openafs.git"
    version: openafs-devel-1_9_1
    dest: openafs

- name: "Build source distribution."
  openafs_build_sdist:
    topdir: openafs
    sdist: openafs/packages

- name: "Build RPM files."
  openafs_build_packages:
    build: all
    sdist: openafs/packages
  register: build_results
```

6.2.5 Return Values

version (always, dict,) OpenAFS and package versions extracted from the source archive.

logfiles (always, list,) The build log files written on the remote node.

packages (always, list,) The list of package files created on the remote node.

6.2.6 Status

- This module is not guaranteed to have a backwards compatible interface. *[preview]*
- This module is maintained by community.

Authors

- Michael Meffie

6.3 openafs_build_redhat_rpms – This module is deprecated. Use openafs_build_packages.

6.3.1 Synopsis

This module is obsolete and will be removed in a future release. Use openafs_build_packages in new playbooks.

6.3.2 Parameters

build (optional, str, all) Specifies which rpms to build.

`all` build source and binary RPMs for userspace and kernel module

`source` build the source RPM only

`userspace` build the source RPM and the userspace RPMs

`modules` build the source RPM and the kmod RPM

sdist (True, path, None) The path on the remote node to the source distribution files directory on the remote node.

The *sdist* directory must contain the `openafs-<version>-src.tar.bz2` source archive and the `openafs-<version>-doc.tar.bz2` documentation archive.

The *sdist* directory may also contain the `ChangeLog` file and the `RELNOTES-<version>` file.

spec (optional, str, None) The path on the remote node to a custom `openafs.spec` file to be used to build the rpm files. The `openafs.spec` file will be extracted from the source archive file when the *spec* option is not provided.

relnotes (optional, str, None) The path on the remote node to a custom `RELNOTES` file to be included in the build.

The `RELNOTES-<version>` in the *sdist* directory will be used when the *relnotes* option is not specified. The `NEWS` file will be extracted from the source archive if the `RELNOTES-<version>` file is not found in the *sdist* directory.

changelog (optional, str, None) The path on the remote node to a custom `ChangeLog` file to be included in the build.

The `ChangeLog` in the *sdist* directory will be used when the `changelog` option is not specified. An empty `ChangeLog` file will be created if the `ChangeLog` is not found in the *sdist* directory,

csdb (optional, path, None) The path on the remote node to a custom `CellServDB` file to be incuded in the build.

The `CellServDB` file in the *sdist* directory will be used when the *csdb* option is not specified. The `CellServDB` file will be extracted from the source archive if the `CellServDB` file is not found in the *sdist* directory.

patchdir (optional, path, I(sdist)) The path on the remote node of the directory containing patch files to be applied.

Patch names are identified by the `PatchXX` directives in the spec file.

kernvers (optional, str, current kernel version) The kernel version to be used when building the kernel module. By default, the kernel version of the running kernel will be used.

topdir (optional, path, C(~/rpmbuild)) The top level rpmbuild workspace directory on the remote node.

logdir (optional, path, I(topdir)/C(BUILD)) The path to write build log files on the remote node.

tar (optional, path, C(tar)) The `tar` program used to unpack the source archive.

tar_extra_options (optional, str, None) Extra command line options to unpack the source archive.

6.3.3 Examples

```
- name: "Checkout OpenAFS source code."
  git:
    repo: "git@openafs.org/openafs.git"
    version: openafs-devel-1_9_1
    dest: openafs

- name: "Build source distribution."
  openafs_build_sdist:
    topdir: openafs
    sdist: openafs/packages

- name: "Build RPM files."
  openafs_build_redhat_rpms:
    build: all
    sdist: openafs/packages
  register: build_results
```

6.3.4 Return Values

version (always, dict,) OpenAFS and package versions extracted from the source archive.

logfiles (always, list,) The build log files written on the remote node.

rpms (always, list,) The list of rpm files created on the remote node.

6.3.5 Status

- This module is not guaranteed to have a backwards compatible interface. *[preview]*
- This module is maintained by community.

Authors

- Michael Meffie

6.4 openafs_build_sdist – Create OpenAFS source distribution archives from a git repo.

6.4.1 Synopsis

Create OpenAFS source and document source distribution archives from a git checkout.

6.4.2 Requirements

The below requirements are needed on the host that executes this module.

- git
- autoconfig
- automake
- libtools
- tar
- gzip
- bzip2
- md5sum
- pod2man

6.4.3 Parameters

sdist (True, path, None) The path on the remote node to write the source distribution files.

This path will be created if it does not exist.

topdir (optional, path, C(openafs)) git project directory on the remote node.

tar (optional, path, detected) `tar` program path

git (optional, path, detected) `git` program path

gzip (optional, path, detected) `gzip` program path

bzip2 (optional, path, detected) `bzip2` program path

md5sum (optional, path, detected) `md5sum` program path

6.4.4 Examples

```
- import_role:
    name: openafs_devel

- name: Checkout source.
  git:
    repo: "git://git.openafs.org/openafs.git"
    version: "openafs-stable-1_8_8"
    dest: "openafs"

- name: Make source distribution files.
  openafs_build_sdist:
    topdir: "openafs"
    sdist: "openafs/packages"
```

6.4.5 Return Values

version (always, dict,) OpenAFS version

files (always, list,) The list of sdist files created on the remote node.

6.4.6 Status

- This module is not guaranteed to have a backwards compatible interface. *[preview]*
- This module is maintained by community.

Authors

- Michael Meffie

6.5 openafs_get_install_paths – Detect installation paths from package installation.

6.5.1 Synopsis

Detect the paths of installed OpenAFS programs and detect configuration directories from installed man pages.

Supports rpm and deb packaging.

6.5.2 Parameters

package_mgr_type (optional, any, autodetect) The package manager type on the node.

Supported values are `rpm` and `apt`.

6.5.3 Examples

```
- name: Get installation paths
  openafs_contrib.openafs.openafs_get_install_paths:
  register: results

- debug:
    msg: >
      Bins are {{ results.bins }}
      Dirs are {{ results.dirs }}
```

6.5.4 Status

- This module is not guaranteed to have a backwards compatible interface. *[preview]*
- This module is maintained by community.

Authors

- Michael Meffie

6.6 openafs_install_bdist – Install OpenAFS binaries built from source

6.6.1 Synopsis

Install OpenAFS binaries built from source code. This module will copy the files in a binary distribution tree to the system directories. Run this module as root. The paths to the installed commands are saved as Ansible local facts.

6.6.2 Parameters

path (True, path, None) Absolute path to the installation file tree to be installed.

exclude (optional, list, None) List of file patterns to be excluded.

6.6.3 Examples

```
- name: Build OpenAFS from source
  openafs_contrib.openafs.openafs_build:
    projectdir: ~/src/openafs
    target: install
    path: /tmp/openafs/bdist

- name: Install OpenAFS binaries as root
  become: yes
  openafs_contrib.openafs.openafs_install:
    path: /tmp/openafs/bdist
    exclude: /usr/vice/etc/*
```

6.6.4 Return Values

msg (always, string, Install completed) Informational message.

files (success, list, [‘/usr/bin/pts’, ‘/usr/sbin/vos’]) Files installed

excluded (success, list, [‘/usr/vice/etc/afs.conf’]) Files excluded from the installation

commands (success, dict, {‘vos’: ‘/usr/sbin/vos’, ‘pts’: ‘/usr/bin/pts’}) Command paths

logfiles (always, list, [‘/tmp/logs/install.log’]) Log files written for troubleshooting

6.6.5 Status

- This module is not guaranteed to have a backwards compatible interface. *[preview]*
- This module is maintained by community.

Authors

- Michael Meffie

6.7 openafs_keys – Add kerberos service keys with asetkey

6.7.1 Synopsis

Import the service keys from a keytab file using the OpenAFS `asetkey` utility.

This module uses `asetkey` rather than the newer `akeyconvert` since `akeyconvert` is not available on all platforms yet.

Before running this module, be sure `asetkey` is installed

The `asetkey` program requires the server `CellServDB` and `ThisCell` files to be present.

A keytab file containing the service keys must be copied to the server.

6.7.2 Parameters

state (False, str, None) c(present) to ensure keys are present in the keyfile(s)

keytab (True, path, None) path to the keytab file on the remote node

cell (True, str, None) AFS cell name

realm (False, str, uppercase of the cell name) Kerberos realm name

asetkey (False, path, Search the local facts, search the path.) asetkey program path

6.7.3 Examples

```
- name: Upload service keytab
  become: yes
  copy:
    src: "files/example.keytab"
    dest: "/usr/afs/etc/rxkad.keytab"
    mode: 0600
    owner: root
    group: root

- name: Add service keys
  become: yes
  openafs_contrib.openafs.openafs_keys:
    state: present
    keytab: /usr/afs/etc/rxkad.keytab
    cell: example.com
```

6.7.4 Return Values

asetkey (success, path,) asetkey path found

have_extended_keys (success, bool,) Indicates if extended keys are supported.

keys (success, list,) keys found in the keytab file

imported (success, list,) Imported key versions

service_principal (success, str,) kerberos service principal

6.7.5 Status

- This module is not guaranteed to have a backwards compatible interface. *[preview]*
- This module is maintained by community.

Authors

- Michael Meffie

6.8 openafs_principal – Create principals and keytab files

6.8.1 Synopsis

Create a kerberos principal on a primary KDC using `kadmin.local` and export the keys to a keytab file on the KDC. The keytab may be transfered to remote nodes with `synchronize` or encrypted with `ansible-vault` then downloaded to the controller for distribution in subsequent plays. This

If the state is `present`, then a principal is added if it is not already present and a keyfile is created. The initial password may be specified with the `password` parameter, otherwise a random key is generated and a keytab file will be created.

If the state is `absent`, then the principal and keytab files are removed if present.

Keytabs for the principals created by the module are stored in the `keytabs` directory on the KDC, readable by root. The default path is `/var/lib/ansible-openafs/keytabs`.

6.8.2 Requirements

The below requirements are needed on the host that executes this module.

- The Kerberos realm has been created.
- `kadmin.local` is installed and in the PATH.

6.8.3 Parameters

state (False, str, present) `present` ensure the principal and keytab file exist.

`absent` ensure the principal and keytab file are removed.

principal (True, str, None) Kerberos principal name.

The name should be provided without the REALM component.

Old kerberos 4 '.' separators are automatically converted to modern '/' separators.

enctypes (False, list, See C(kadmin)) Kerberos encryption and salt types.

See `kadmin` documenation for possible values.

acl (False, str, None) Administrative permissions

keytab_name (optional, str, principal name with '/' characters replaced by '.' characters.) Alternative keytab name.

keytabs (False, path, C(/var/lib/ansible-openafs/keytabs))

kadmin (False, path, search PATH)

6.8.4 Examples

```
- name: Create an AFS service key
  become: yes
  openafs_contrib.openafs.openafs_principal:
    principal: afs/example.com
    encryption_types:
      - aes128-cts:normal
      - aes256-cts:normal
  register: service_key

- name: Download the keytab to controller for distribution
  become: yes
  fetch:
    flat: yes
    src: "{{ service_key.keytab }}"
    dest: "rxkad.keytab"

# Requires an old version of Kerberos.
- name: Obsolete DES key for testing
  become: yes
  openafs_contrib.openafs.openafs_principal:
    state: present
    service: afs
    principal: afs/broken.com
    enctype: des-cbc-crc:afs3

- name: Create some user principals
  become: yes
  openafs_contrib.openafs.openafs_principal:
    state: present
    principal: "{{ item }}"
    password: "{{ initial_password }}"
  with_items:
    - alice
    - bob
    - charlie
```

6.8.5 Return Values

attributes (success, list,) Principal attributes from `get_principal`

debug (always, list,) kadmin commands executed and output

kadmin (always, path,) kadmin executable path

keytab (success, path,) Path of the generated keytab on the remote node.

principal (success, str,) principal name

realm (, str,) realm name

6.8.6 Status

- This module is not guaranteed to have a backwards compatible interface. *[preview]*
- This module is maintained by community.

Authors

- Michael Meffie

6.9 openafs_selinux_module – Create and install an selinux module from input files

6.9.1 Synopsis

Build the selinux module from the given input files.

6.9.2 Parameters

state (optional, any, None) c(present) is currently the only supported state.

name (optional, any, openafs) name of the selinux module

path (optional, any, /var/lib/ansible-openafs/selinux) Path to the Type Enforcement (te) and File Context (fc) input files and the destination path of the output pp and mod files.

6.9.3 Examples

```
- name: Copy the SELinux module definitions for openafs server
  become: yes
  template:
    dest: "/var/lib/ansible-openafs/selinux/{{ item }}"
    src: "{{ role_path }}/templates/{{ item }}.j2"
  with_items:
    - openafs.te
    - openafs.fc

- name: Build SELinux module for openafs server
  become: yes
  openafs_contrib.openafs.openafs_selinux_module:
    name: openafs
    path: /var/lib/ansible-openafs/selinux
```

6.9.4 Return Values

module (success, str,) Path to the module

version (success, str,) Module version

6.9.5 Status

- This module is not guaranteed to have a backwards compatible interface. *[preview]*
- This module is maintained by community.

Authors

- Michael Meffie

6.10 openafs_selinux_relabel – Relabel selinux context for server files

6.10.1 Synopsis

Relabel the server directories after the files have been installed and the configuration files updated.

Relabel the partition directories and the AlwaysAttach file, when present.

Safe the list of directories relabelled in the openafs local facts file to avoid running restorecon on subsequent plays.

6.10.2 Examples

```
- name: Relabel
  become: yes
  openafs_contrib.openafs.openafs_selinux_relabel:
```

6.10.3 Status

- This module is not guaranteed to have a backwards compatible interface. *[preview]*
- This module is maintained by community.

Authors

- Michael Meffie

6.11 openafs_service_properties – Manage Solaris service properties.

6.11.1 Synopsis

Set service properties using the Solaris svc commands.

6.11.2 Parameters

state (optional, any, None) If `present` the `property` will be set to the given `value`.

If `absent` the `property` will be deleted.

service (True, any, None) The SMF service name.

instance (optional, any, default) The SMF service instance name.

property (True, any, None) The property name.

value (optional, any, None) The property value. Requiried when `state` is `present`.

single (optional, bool, True) If True, the value is a single string.

If False, the value is a space separated list of strings.

6.11.3 Examples

```
- name: Set client startup arguments.
  openafs_service_property:
    state: present
    service: network/openafs/client
    instance: default
    property: afsd/arguments
    value: -dynroot -fakestat -afsd
    single: no
```

6.11.4 Return Values

service (always, str,) SMF service name

instance (always, str,) SMF service instance

property (always, str,) SMF service property name

value (on success, str,) SMF service property value

single (always, bool,) Value is a single string

6.11.5 Status

- This module is not guaranteed to have a backwards compatible interface. *[preview]*
- This module is maintained by community.

Authors

- Michael Meffie

6.12 openafs_store_facts – Store OpenAFS facts in a json file

6.12.1 Synopsis

Store facts in the json formatted c(openafs.fact) file on the remote host. This file is located in the c(/etc/ansible/facts.d) directory or the path specified by the c(factsdir) parameter. The c(openafs.fact) file is read by Ansible when facts are gathered on subsequent plays.

The c(openafs.fact) contains a dictionary of facts which can be accessed from c(ansible_local.openafs).

6.12.2 Parameters

state (optional, any, None) c(update) update the facts

factsdir (optional, path, /etc/ansible/facts.d) Path to the c(openafs.fact) file

6.12.3 Examples

6.12.4 Status

- This module is not guaranteed to have a backwards compatible interface. *[preview]*
- This module is maintained by community.

Authors

- Michael Meffie

6.13 openafs_user – Create an OpenAFS user

6.13.1 Synopsis

Create or remove a user.

Optionally create new groups and add the user to groups.

Localauth authentication may be used on server nodes, running as root.

Keytab based authentication may be used on client nodes. This requires a keytab for a user in the system:adminstrators group and a member of the UserList on all of the database servers.

6.13.2 Parameters

state (optional, str, present) `present` create user and groups when not present

`absent` remove user when not present

user (True, str, None) The OpenAFS username.

id (False, int, 0) The OpenAFS pts id.

The next available id will be selected if omitted or 0.

groups (False, list, None) The OpenAFS group names the user is a member.

Non-system groups will be created.

localauth (optional, bool, False) Indicates if the `-localauth` option is to be used for authentication.

This option should only be used when running on a server.

auth_user (optional, str, admin) The afs user name to be used when `localauth` is False.

The user must be a member of the `system:administrators` group and must be a server superuser, that is, set in the `UserList` file on each server in the cell.

Old kerberos 4 '.' separators are automatically converted to modern '/' separators.

This option may only be used if a client is installed on the remote node.

auth_keytab (optional, str, admin.keytab) The path on the remote host to the keytab file to be used to authenticate.

The keytab file must already be present on the remote host.

This option may only be used if a client is installed on the remote node.

6.13.3 Examples

```
- name: Create users
  openafs_contrib.openafs.openafs_user:
     name: "{{ item }}"
     group: tester
  with_items:
    - alice
    - bob
    - charlie
```

6.13.4 Return Values

user (, dictionary,) User information.

6.13.5 Status

- This module is not guaranteed to have a backwards compatible interface. *[preview]*
- This module is maintained by community.

Authors

- Michael Meffie

6.14 openafs_volume – Create an OpenAFS volume

6.14.1 Synopsis

Create or remove a volume.

Optionally, create read-only volumes, and release the volume.

Optionally, mount the volume and set the ACL rigths in the filespace.

Volume mounting requires a client running on the remote node.

Localauth authentication may be used on server nodes, running as root. When running in this mode, volumes maybe created, but not mounted.

Keytab based authentication may be used on client nodes to mount volumes and set root directory ACLs. This requires a keytab for a user in the system:adminstrators group and a member of the UserList on all of the servers.

6.14.2 Parameters

state (False, str, <present>) `present` ensure the volume is present, `absent` ensure the volume is removed

volume (True, str, None) Name of the read-write volume.

server (optional, str, first fileserver entry found in VLDB) The initial volume fileserver location.

If provided, should be the hostname or fileserver address.

If not provided, the first fileserver address from `vos listaddrs` will be used.

The volume will not be moved if it already exists on a different server.

This option is ignored when the state is `absent`.

partition (optional, str, the first partition found on the fileserver) The initial volume partition id.

If provided, should be the partition id; `'a'` .. `'iu'`.

If not provided, the first partition found from `vos listpart` will be used.

The volume will not be moved if it already exists on a different partition.

This option is ignored when the state is `absent`.

mount (False, str, None) The initial mount point path.

Should be the fully-qualified path to the mount point to be created.

The read/write path variant will be used if it is available.

A read/write mount point will also be created for the `root.cell` volume.

The `i` and `a` ACL rights will be temporarily assigned to the mount point parent directory in order to create the mount point if those rights are missing.

The volume containing the parent volume will be released if a mount point was created.

The volume will be created but not mounted if the `mount` option is not given.

This option is ignored when the state is `absent`.

This option may only be used if a client is installed on the remote node.

acl (False, str, None) The access control list to be set in the volumes root directory.

The `acl` option my be specified as a list of strings. Each string contains a pair of strings separated by a space. The substring names a user or group, the second indicates the access rights.

See `fs setacl` for details.

This option may only be used if a client is installed on the remote node.

quota (False, int, 0) The initial volume quota.

replicas (False, int, 0) The number of read-only volumes to be created, including the read-only clone on the same fileserver and partition as the read/write volume.

The `replicas` option indicates the minumum number of read-only volumes desired.

localauth (optional, bool, False) Indicates if the `-localauth` option is to be used for authentication.

This option should only be used when running on a server.

The `mount` and `acl` options may not be used with `localauth`.

auth_user (optional, str, admin) The afs user name to be used when `localauth` is False.

The user must be a member of the `system:administrators` group and must be a server superuser, that is, set in the `UserList` file on each server in the cell.

Old kerberos 4 '.' separators are automatically converted to modern '/' separators.

This option may only be used if a client is installed on the remote node.

auth_keytab (optional, str, admin.keytab) The path on the remote host to the keytab file to be used to authenticate.

The keytab file must already be present on the remote host.

This option may only be used if a client is installed on the remote node.

6.14.3 Examples

```
- name: Create afs root volume
  openafs_contrib.openafs.openafs_volume:
    state: present
    name: root.afs
    mount: /afs
    acl: "system:anyuser read"
    replicas: 3

- name: Create cell root volume
  openafs_contrib.openafs.openafs_volume:
    state: present
    name: root.cell
    mount: /afs/example.com
    acl: "system:anyuser read"
    replicas: 3

- name: Create a volume
  openafs_contrib.openafs.openafs_volume:
    state: present
    name: test
    mount: /afs/example.com/test
    acl:
      - "bob all"
      - "system:anyuser read"
      - "system:authuser write"

- name: Alternate acl format
  openafs_contrib.openafs.openafs_volume:
    state: present
    name: test
    mount: /afs/example.com/test
    acl:
      bob: all
      "system:anyuser": read
      "system:authuser": write
```

6.14.4 Return Values

acl (success, list,) List of acl strings set in the volume root directory

mount (success, str,) Mount point path

volume (success, dict,) Volume information

6.14.5 Status

- This module is not guaranteed to have a backwards compatible interface. *[preview]*
- This module is maintained by community.

Authors

- Michael Meffie

6.15 openafs_wait_for_quorum – Wait for the dbserver connection and quorum

6.15.1 Synopsis

Wait until the VLDB and PRDB database elections are completed and a sync site is set.

6.15.2 Examples

```
- name: Wait for database quorum
  become: yes
  openafs_contrib.openafs.openafs_wait_for_quorum:
    sleep: 10
    timeout: 600
  when:
    - afs_is_dbserver
```

6.15.3 Status

- This module is not guaranteed to have a backwards compatible interface. *[preview]*
- This module is maintained by community.

Authors

- Michael Meffie

6.16 openafs_wait_for_registration – Wait for the fileserver VLDB registration

6.16.1 Synopsis

Wait for the fileserver VLDB registration to be completed.

6.16.2 Parameters

timeout (optional, int, 600) Maximum time to wait in seconds.

delay (optional, int, 0) Number of seconds to delay before waiting.

sleep (optional, int, 20) Number of seconds to wait between retries.

signal (optional, bool, True) If true, issue a XCPU signal to the fileserver to force it to resend the VLDB registration after `sleep` seconds has expired.

By default, the fileserver will retry the VLDB registration every 5 minutes untill the registration succeeds. This option can be used to force the retry to happen sooner. As a side-effect, XCPU signal will trigger a dump of the fileserver hosts and callback tables, so this option must be used with caution.

6.16.3 Examples

```
- name: Wait for fileserver registration
  openafs_contrib.openafs.openafs_wait_for_registration:
    sleep: 10
    timeout: 600
    signal: no
  when:
    - afs_is_fileserver
```

6.16.4 Status

- This module is not guaranteed to have a backwards compatible interface. *[preview]*
- This module is maintained by community.

Authors

- Michael Meffie

CHAPTER

SEVEN

PLUGINS

7.1 Lookup

7.1.1 counter – Increment named integer counters.

Synopsis

Counter values are saved in a json file on the controller. The default location of the counter file is `~/.ansible/counter.json` Set the `ANSIBLE_OPENAFS_COUNTER_DIR` environment variable to specify an alternate location.

By default, the the *counter* lookup plugin increments then returns the incremented counter value.

Specify the `,current` suffix on the counter name to retrieve the current counter value without incrementing the counter.

Specify the `,reset` suffix on the counter name to reset the counter value to zero.

File locking used for mutual exclusion in case more than one playbook is running at a time.

Parameters

_terms (True, any, None) list of counter names, in the form *<name>[,<operation>]*, where *<name>* is the counter name and *<operation>* is one of `next`, `current`, `reset`

The default operation is `next`

Examples

```
- name: "Lookup the current value of test_a."
  debug:
    msg: "{{ lookup('openafs_contrib.openafs.counter', 'test_a,current') }}"

- name: "Increment counter test_a."
  debug:
    msg: "{{ lookup('openafs_contrib.openafs.counter', 'test_a') }}"

- name: "Increment counters using 'with_' syntax."
  debug:
    var: item
  with_openafs_contrib.openafs.counter:
    - test_a
    - test_b
    - test_c

- name: "Reset counters using 'with_' syntax."
  assert:
    that: item == 0
  with_openafs_contrib.openafs.counter:
    - test_a,reset
    - test_b,reset
    - test_c,reset
```

Return Values

_list (, list,) List of counter values.

Status

Authors

- Michael Meffie

CHAPTER

EIGHT

PLATFORM NOTES

8.1 Red Hat Enterprise Linux/CentOS

SE Linux `enforcing` mode is only supported when OpenAFS is installed from RPM packages. Be sure to first set the SE Linux mode to `permissive` when installing OpenAFS from source or a binary distribution.

The OpenAFS kernel module may be installed with DKMS or from a pre-built kernel module package. The DKMS system will automatically rebuild the kernel module when the linux kernel is updated, but requires the module to be rebuilt on each node and requires the kernel-devel package version to match the running kernel version to be installed on the nodes. The prebuilt kernel package must match the version of the running kernel, so requires a repository which is updated for each kernel version.

Set the host variable `afs_module_install_method` to `dkms` to install a client with DKMS, and to `kmod` to install the client with a prebuilt kernel module package.

8.2 Solaris

Historically, Transarc-style binary distributions are used to install OpenAFS on Solaris. Please set `afs_install_method` on Solaris nodes to `bdist` or `source` to install from a binary distribution or from source code. The `openafs_build` module will build a Transarc-style binary distribution by default on Solaris nodes.

OpenAFS client and servers are started with legacy style `sysv` init scripts on Solaris.

CHAPTER

NINE

LICENSE